

Parameterized complexity results for 1-safe Petri nets

M. Praveen and Kamal Lodaya

The Institute of Mathematical Sciences, Chennai 600113, India

Abstract. We associate a graph with a 1-safe Petri net and study the parameterized complexity of various problems with parameters derived from the graph. With treewidth as the parameter, we give W[1]-hardness results for many problems about 1-safe Petri nets. As a corollary, this proves a conjecture of Downey et. al. about the hardness of some graph pebbling problems. We consider the parameter benefit depth (that is known to be helpful in getting better algorithms for general Petri nets) and again give W[1]-hardness results for various problems on 1-safe Petri nets. We also consider the stronger parameter vertex cover number. Combining the well known automata-theoretic method and a powerful fixed parameter tractability (FPT) result about Integer Linear Programming, we give a FPT algorithm for model checking Monadic Second Order (MSO) formulas on 1-safe Petri nets, with parameters vertex cover number and the size of the formula.

1 Introduction

Petri nets are popular for modelling because they offer a succinct representation of loosely coupled communicating systems. Some powerful techniques are available but the complexity of analysis is high. In his lucid survey [8], Esparza summarizes the situation as follows: almost every interesting analysis question on the behaviour of general Petri nets is EXPSPACE-hard, and almost every interesting analysis question on the behaviour of 1-safe Petri nets is PSPACE-hard. By considering special subclasses of nets slightly better results can be obtained. Esparza points out that T-systems (also called marked graphs) and S-systems (essentially sequential transition systems) are the largest subclasses where polynomial time algorithms are available. We therefore look for a *structural parameter* with respect to which some analysis problems remain tractable.

Parameterized complexity. A brief review will not be out of place here. Let Σ be a finite alphabet in which instances $I \in \Sigma^*$ of a problem $\Pi \subseteq \Sigma^*$ are specified, where Π is the set of YES instances. The complexity of a problem is stated in terms of the amount of resources—space, time—needed by any algorithm solving it, measured as a function of the size $|I|$ of the problem instance. In parameterized complexity, introduced by Downey and Fellows [5], the dependence of resources needed is also measured in terms of a parameter $\kappa(I)$ of the input, which is usually less than the input size $|I|$. A parameterized problem is said

to be **fixed parameter tractable** (FPT) if it can be solved by an algorithm with running time $f(\kappa(I))poly(|I|)$ where f is some computable function and $poly$ is a polynomial. (Similarly, a PARAPSPACE algorithm [10] is one that runs in space $f(\kappa(I))poly(|I|)$.)

For example, consider the problem of checking that all strings accepted by a given finite state automaton satisfy a given Monadic Second Order (MSO) sentence. The size of an instance of this problem is the sum of sizes of the automaton and the MSO sentence. If the size of the MSO sentence is considered as a parameter, then this problem is FPT, by Büchi, Elgot, Trakhtenbrot theorem [2].

There is a parameterized complexity class $W[1]$, lowest in a hierarchy of intractable classes called the W-hierarchy [5] (similar to the polynomial time hierarchy). A parameterized problem complete for $W[1]$ is to decide if there is an accepting computation of at most k steps in a given non-deterministic Turing machine, where the parameter is k [5]. It is widely believed that parameterized problems hard for $W[1]$ are not FPT. To prove that a problem is hard for a parameterized complexity class, we have to give a **parameterized reduction** from a problem already known to be hard to our problem. A parameterized reduction from (Π, κ) to (Π', κ') is an algorithm A that maps problem instances in (resp. outside) Π to problem instances in (resp. outside) Π' . There must be computable functions f and g and a polynomial p such that the algorithm A on input I terminates in time $f(\kappa(I))p(|I|)$ and $\kappa'(A(I)) \leq g(\kappa(I))$, where $A(I)$ is the problem instance output by A .

Results. Demri, Laroussinie and Schnoebelen considered synchronized transition systems, a form of 1-safe Petri nets [4] and showed that the number of synchronizing components (processes) is not a parameter which makes analysis tractable. Likewise, our first results are negative. All parameters mentioned below are defined in Sect. 2.

- With the pathwidth of the flow graph of the 1-safe Petri net as parameter, reachability, coverability, Computational Tree Logic (CTL) and the complement of Linear Temporal Logic (LTL) model checking problems are all $W[1]$ -hard, even when the size of the formula is a constant. In contrast, for the class of sequential transition systems and formula size as parameter, Büchi’s theorem is that model checking for MSO logic is FPT.
- As a corollary, we also prove a conjecture of Downey, Fellows and Stege that the SIGNED DIGRAPH PEBBLING problem [6, section 5] is $W[1]$ -hard when parameterized by treewidth.
- With the benefit depth of the 1-safe Petri net as parameter, reachability, coverability, CTL and the complement of LTL model checking problems are $W[1]$ -hard, even when the size of the formula is a constant.

We are luckier with our third parameter.

- With the vertex cover number of the flow graph and formula size as parameters, MSO model checking is FPT.

Perspective. As can be expected from the negative results, the class of 1-safe Petri nets which are amenable to efficient analysis (i.e., those with small vertex cover) is not too large. But even for this class, a reachability graph construction can be of exponential size, so just an appeal to Büchi’s theorem is not sufficient to yield our result.

Roughly speaking, our FPT algorithm works well for systems which have a small “core” (vertex cover), a small number of “interface types” with this core, but any number of component processes using these interface types to interact with the core (see Fig. 9). Thus, we can have a large amount of conflict and concurrency but a limited amount of causality. Recall that S-systems and T-systems have no concurrency and no conflict, respectively. Since all we need from the logic is a procedure which produces an automaton from a formula, we are able to use the most powerful, MSO logic. Our proofs combine the well known automata-theoretic method [2, 22, 12] with a powerful result about feasibility of Integer Linear Programming (ILP) parameterized by the number of variables [14, 13, 11].

Related work. Drusinsky and Harel studied nondeterminism, alternation and concurrency in finite automata from a complexity point of view [7]. Their results also hold for 1-bounded Petri nets.

The SIGNED DIGRAPH PEBBLING problem considered by Downey, Fellows and Stege [6] can simulate Petri nets. They showed that with treewidth and the length of the firing sequence as parameters, the reachability problem is FPT. They conjectured that with treewidth alone as parameter, the problem is W[1]-hard.

Fellows *et al* showed that various graph layout problems that are hard with treewidth as parameter (or whose complexity parameterized by treewidth is not known) are FPT when parameterized by vertex cover number [9]. They also used tractability of ILP and extended feasibility to optimization.

Acknowledgements. We thank the anonymous CONCUR referees for providing detailed comments that helped in improving the presentation.

2 Preliminaries

2.1 Petri nets

A Petri net is a 4-tuple $\mathcal{N} = (P, T, Pre, Post)$, P a set of places, T a set of transitions, $Pre : P \times T \rightarrow \{0, 1\}$ (arcs going from places to transitions) and $Post : P \times T \rightarrow \{0, 1\}$ (arcs going from transitions to places) the incidence functions. A place p is an **input** (**output**) place of a transition t if $Pre(p, t) = 1$ ($Post(p, t) = 1$) respectively. We use $\bullet t$ ($t \bullet$) to denote the set of input (output) places of a transition t . In diagrams, places are shown as circles and transitions as thick bars. Arcs are shown as directed edges between places and transitions.

Given a Petri net \mathcal{N} , we associate with it an undirected **flow graph** $G(\mathcal{N}) = (P, E)$ where $(p_1, p_2) \in E$ iff for some transition t , $Pre(p_1, t) + Post(p_2, t) \geq 1$

and $Pre(p_2, t) + Post(p_2, t) \geq 1$. If a place p is both an input and an output place of some transition, the vertex corresponding to p has a self loop in $G(\mathcal{N})$.

A **marking** $M : P \rightarrow \mathbb{N}$ can be thought of as a configuration of the Petri net, with each place p having $M(p)$ tokens. We will only deal with **1-safe** Petri nets in this paper, where the range of markings is restricted to $\{0, 1\}$. Given a Petri net \mathcal{N} with a marking M and a transition t such that for every place p , $M(p) \geq Pre(p, t)$, the transition t is said to be **enabled** at M and can be **fired** (denoted $M \xrightarrow{t} M'$) giving $M'(p) = M(p) - Pre(p, t) + Post(p, t)$ for every place p . This is generalized to a **firing sequence** $M \xRightarrow{t_1} M_1 \xRightarrow{t_2} \dots \xRightarrow{t_r} M_r$, more briefly $M \xRightarrow{t_1 t_2 \dots t_r} M_r$. A firing sequence ρ enabled at M_0 is said to be **maximal** if it is infinite, or if $M_0 \xRightarrow{\rho} M$ and no transition is enabled at M .

Definition 1 (Reachability, coverability). *Given a 1-safe Petri net \mathcal{N} with initial marking M_0 and a target marking $M : P \rightarrow \{0, 1\}$, the reachability problem is to decide if there is a firing sequence ρ such that $M_0 \xRightarrow{\rho} M$. The coverability problem is to decide if there is a firing sequence ρ and some marking $M' : P \rightarrow \{0, 1\}$ such that $M_0 \xRightarrow{\rho} M'$ and $M'(p) \geq M(p)$ for every place p .*

2.2 Logics

Linear Temporal Logic (LTL) is a formalism in which many properties of transition systems can be specified [8, section 4.1]. We use the syntax of [8], in particular the places P are the atomic formulae. The LTL formulas are interpreted on **runs**, sequences of markings $\pi = M_0 M_1 \dots$ from a firing sequence of a 1-safe Petri net. The satisfaction of a LTL formula ϕ at some position j in a run is defined inductively, in particular $\pi, j \models p$ iff $M_j(p) = 1$. Much more expressive is the Monadic Second Order (MSO) logic of Büchi [2], interpreted on a maximal run $M_0 M_1 \dots$, with $\pi, s \models p(x)$ iff $M_{s(x)}(p) = 1$ under an assignment s to the variables. Boolean operations, first-order and monadic second-order quantifiers are available as usual.

Computational Tree Logic (CTL) is another logic that can be used to specify properties of 1-safe Petri nets. The reader is referred to [8, section 4.2] for details.

Definition 2 (Model checking). *Given a 1-safe Petri net \mathcal{N} with initial marking M_0 and a logical formula ϕ , the model checking problem (for that logic) is to decide if for every maximal firing sequence ρ , the corresponding maximal run π satisfies $\pi, 0 \models \phi$.*

Reachability, coverability and LTL model checking for 1-safe Petri nets are all PSPACE-complete [8]. Habermehl gave an automata-theoretic model checking procedure for Linear Time μ -calculus on general Petri nets [12].

2.3 Parameters

The study of parameterized complexity derived an initial motivation from the study of graph parameters. Many NP-complete problems can be solved in polynomial time on trees and are FPT on graphs that have tree-structured decompositions.

Definition 3 (Tree decomposition, treewidth, pathwidth). A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\mathcal{T}, (B_\tau)_{\tau \in \text{nodes}(\mathcal{T})})$, where \mathcal{T} is a tree and $(B_\tau)_{\tau \in \text{nodes}(\mathcal{T})}$ is a family of subsets of V such that:

- For all $v \in V$, the set $\{\tau \in \text{nodes}(\mathcal{T}) \mid v \in B_\tau\}$ is nonempty and connected in \mathcal{T} .
- For every edge $(v_1, v_2) \in E$, there is a $\tau \in \text{nodes}(\mathcal{T})$ such that $v_1, v_2 \in B_\tau$.

The *width* of such a decomposition is the number $\max\{|B_\tau| \mid \tau \in \text{nodes}(\mathcal{T})\} - 1$. The **treewidth** $\text{tw}(G)$ of G is the minimum of the widths of all tree decompositions of G . If the tree \mathcal{T} in the definition of tree decomposition is a path, we get a *path decomposition*. The **pathwidth** $\text{pw}(G)$ of G is the minimum of the widths of all path decompositions of G .

From the definition, it is clear that pathwidth is at least as large as treewidth and any problem that is $W[1]$ -hard with pathwidth as parameter is also $W[1]$ -hard with treewidth as parameter. A fundamental result by Courcelle [3] shows that graphs of small treewidth are easier to handle algorithmically: checking whether a graph satisfies a MSO sentence is FPT if the graph's treewidth and the MSO sentence's length are parameters. In our context, the state space of a concurrent system can be considered a graph. However, due to the state explosion problem, the state space can be very large. Instead, we impose treewidth restriction on a compact representation of the large state space — a 1-safe Petri net. Note also that we are not model checking the state space itself but only the language of words generated by the Petri net.

Definition 4 (Vertex cover number). A *vertex cover* $VC \subseteq V$ of a graph $G = (V, E)$ is a subset of vertices such that for every edge in E , at least one of its vertices is in VC . The **vertex cover number** of G is the size of a smallest vertex cover.

Definition 5 (Benefit depth [18]). The set of places $\text{ben}(p)$ benefited by a place p is the smallest set of places (including p) such that any output place of any output transition of a place in $\text{ben}(p)$ is also in $\text{ben}(p)$. The **benefit depth** of a Petri net is defined as $\max_{p \in P} \{|\text{ben}(p)|\}$.

Benefit depth can be thought of as a generalization of the out-degree in directed graphs. For a Petri net, we take vertex covers of its flow graph $G(\mathcal{N})$. Any vertex cover of $G(\mathcal{N})$ should include all vertices that have self loops. It was shown in [18, 17] that benefit depth and vertex cover number bring down the complexity of coverability and boundedness in general Petri nets from exponential space-complete [19] to PARAPSPACE.

3 Lower bounds for 1-safe Petri nets and pebbling

3.1 1-safe Petri nets, treewidth and pathwidth

Here we prove $W[1]$ -hardness of reachability in 1-safe Petri nets with the pathwidth of the flow graph as parameter, through a parameterized reduction from

the parameterized Partitioned Weighted Satisfiability (p-PW-SAT) problem. The **primal graph** of a propositional CNF formula has one vertex for each propositional variable, and an edge between two variables iff they occur together in a clause. An instance of p-PW-SAT problem is a triple $(\mathcal{F}, \text{part} : \Phi \rightarrow \{1, \dots, k\}, \text{tg} : \{1, \dots, k\} \rightarrow \mathbb{N})$, where \mathcal{F} is a propositional CNF formula, part partitions the set of propositional variables Φ into k parts and we need to check if there is a satisfying assignment that sets exactly $\text{tg}(r)$ variables to \top in each part r . Parameters are k and the pathwidth of the primal graph of \mathcal{F} . We showed in an earlier paper that p-PW-SAT is $W[1]$ -hard when parameterized by the number of parts k and the pathwidth of the primal graph [16, Lemma 6.1].

Now we will demonstrate a parameterized reduction from p-PW-SAT to reachability in 1-safe Petri nets, with the pathwidth of the flow graph as parameter. Given an instance of p-PW-SAT, let q_1, \dots, q_n be the variables used. Construct an optimal path decomposition of the primal graph of the CNF formula in the given p-PW-SAT instance (doing this is FPT [1]). For every clause in the CNF formula, the primal graph contains a clique formed by all variables occurring in that clause. There will be at least one bag in the path decomposition of the primal graph that contains all vertices in this clique [5, Lemma 6.49]. Order the bags of the path decomposition from left to right and call the clause whose clique appears first C_1 , the clause whose clique appears second as C_2 and so on. If more than one such clique appear for the first time in the same bag, order the corresponding clauses arbitrarily. Let C_1, \dots, C_m be the clauses ordered in this way. We will call this the **path decomposition ordering** of clauses, and use it to prove that the pathwidth of the flow graph of the constructed 1-safe Petri net is low (Lemma 7). For a partition r between 1 and k , we let $n[r]$ be the number of variables in r . Following are the places of our 1-safe Petri net.

1. For every propositional variable q_i used in the given p-PW-SAT instance, places q_i, x_i, \bar{x}_i .
2. For every partition r between 1 and k , places $t\uparrow^r, f\uparrow^r, tu_r^0, \dots, tu_r^{\text{tg}(r)}$ and $fl_r^0, \dots, fl_r^{n[r]-\text{tg}(r)}$.
3. For each clause C_j , a place C_j . Additional places C_{m+1}, s, g .

The construction of the Petri net is illustrated in the following diagrams. The notation $\text{part}(i)$ stands for the partition to which q_i belongs. Intuitively, the truth assignment of q_i is determined by firing t_i or f_i in Fig. 1. The token in x_i/\bar{x}_i is used to check satisfaction of clauses later. The token in $t\uparrow^{\text{part}(i)}/f\uparrow^{\text{part}(i)}$ is used to count the number of variables set to \top/\perp in each part, with the part of the net in Fig. 2. For each clause C_j between 1 and m , the part of the net shown in Fig. 3 is constructed. In Fig. 3, it is assumed that $C_j = q_1 \vee \bar{q}_2 \vee q_3$. Intuitively, a token can be moved from place C_j to C_{j+1} iff the clause C_j is satisfied by the truth assignment determined by the firings of t_i/f_i for each i between 1 and n . The net in Fig. 4 checks that the target has been met in all partitions.

The initial marking of the constructed net consists of 1 token each in the places $q_1, \dots, q_n, s, tu_1^0, \dots, tu_k^0, fl_1^0, \dots, fl_k^0$ and C_1 , with 0 tokens in all other places. The final marking to be reached has a token in the places s and g .

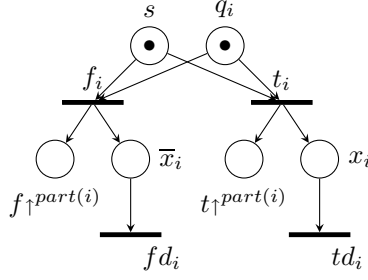


Fig. 1. Part of the net for each variable q_i

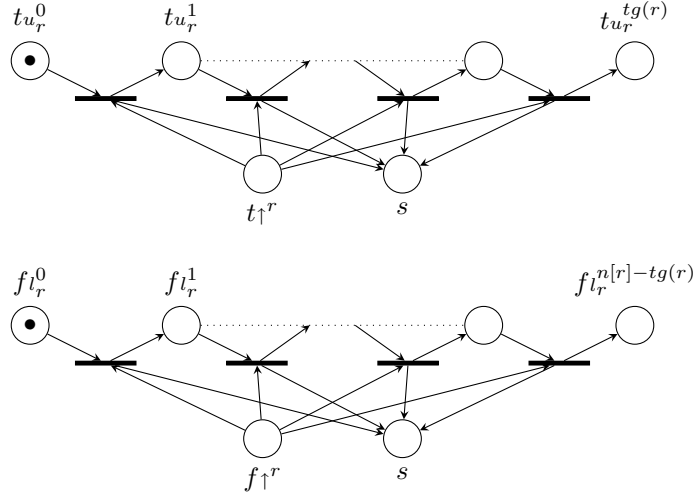


Fig. 2. Part of the net for each part r between 1 and k

Lemma 6. *Given a p-PW-SAT instance, constructing the Petri net described above is FPT. The constructed Petri net is 1-safe. The given instance of p-PW-SAT is a YES instance iff in the constructed 1-safe net, the required final marking can be reached from the given initial marking.*

Proof. The only non-trivial process in the construction of the Petri net is computing an optimal path decomposition of the primal graph of the CNF formula in the given p-PW-SAT instance. Doing this is FPT and the rest of the construction can be done in polynomial time. It is not difficult to see that the constructed net is 1-safe.

Suppose the given instance of p-PW-SAT is a YES instance. Starting with $i = 1$, repeat the following firing sequence for each i between 1 and n . If q_i is \top in the witnessing satisfying assignment, fire t_i else fire f_i . Then use the

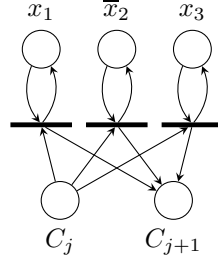


Fig. 3. Part of the net for each clause C_j

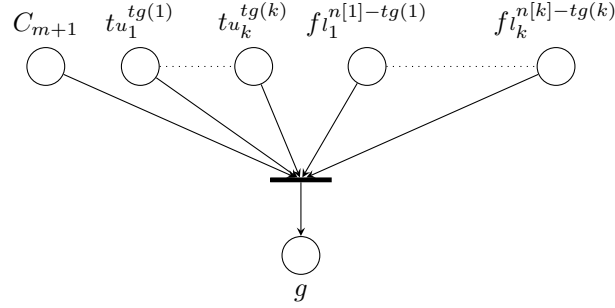


Fig. 4. Part of the net to check that target has been met

token thus put into $t\uparrow^{part(i)}/f\uparrow^{part(i)}$ respectively to shift a token one place to the right in Fig. 2 and put a token back in the place s . Continue with the next i . Since the witnessing assignment meets the target in each partition, we will have one token each in the places $tu_1^{tg(1)}, \dots, tu_k^{tg(k)}, fl_1^{n[1]-tg(1)}, \dots, fl_k^{n[k]-tg(k)}$. In addition, there will be a token in x_i/\bar{x}_i iff the witnessing assignment set q_i to \top/\perp respectively. Since this witnessing assignment satisfies all the clauses of the CNF formula, we can move the initial token in C_1 to C_{m+1} using the transitions in Fig. 3. Now, the transition in Fig. 4 can be fired to get a token into the place g . Now, the only tokens left are those in the places s and g , and those in x_i/\bar{x}_i . We can remove the tokens in x_i/\bar{x}_i by firing td_i/fd_i to reach the final marking.

Suppose the required final marking is reachable in the constructed Petri net. Since a token has to be added to the place g to reach the final marking and the transition in Fig. 4 is the only transition that can add tokens to g , all input places of that transition must receive a token. The only way to get a token in places $tu_r^{tg(r)}$ is to shift the initial token in the place tu_r^0 $tg(r)$ times. This requires exactly $tg(r)$ tokens in the place $t\uparrow^r$. A similar argument holds for getting a token in $fl_r^{n[r]-tg(r)}$. Since the only way to add a token to $t\uparrow^r/f\uparrow^r$ is to fire transitions t_i/f_i (such that $part(i) = r$), the only way to get a token

each in $t_{u_1}^{tg(1)}, \dots, t_{u_k}^{tg(k)}, f_{l_1}^{n[1]-tg(1)}, \dots, f_{l_k}^{n[k]-tg(k)}$ is to fire either t_i or f_i for each i between 1 and n . Consider any firing sequence reaching the required final marking. Consider the truth assignment to q_1, \dots, q_n that assigns \top to exactly those variables q_i such that t_i was fired in the firing sequence. This truth assignment meets the target for each part since this firing sequence adds one token each to the places $t_{u_1}^{tg(1)}, \dots, t_{u_k}^{tg(k)}, f_{l_1}^{n[1]-tg(1)}, \dots, f_{l_k}^{n[k]-tg(k)}$. To reach the final marking, a token is also required at the place C_{m+1} . The only way to get this token is to shift the initial token in C_1 to C_{m+1} through the transitions in Fig. 3. This means that every clause is satisfied by the truth assignment we constructed. \square

It remains to prove that the pathwidth of the flow graph of the constructed 1-safe net is a function of the parameters of the p-PW-SAT instance.

Lemma 7. *Suppose a given instance of p-PW-SAT has a CNF formula whose primal graph has pathwidth p_w and k parts. Then, the flow graph of the 1-safe net constructed as described above has pathwidth at most $3p_w + 4k + 7$.*

Proof. We show a path decomposition of the flow graph of the net. Call the set of places $\{s, g, C_{m+1}, t_{\uparrow}^1, \dots, t_{\uparrow}^k, f_{\uparrow}^1, \dots, f_{\uparrow}^k, t_{u_1}^{tg(1)}, \dots, t_{u_k}^{tg(k)}, f_{l_1}^{n[1]-tg(1)}, \dots, f_{l_k}^{n[k]-tg(k)}\}$ as P_1 . Consider an optimal path decomposition of the primal graph of the CNF formula. In every bag, replace every occurrence of each q_i by the set $\{q_i, x_i, \bar{x}_i\} \cup P_1$.

Let C_1, \dots, C_m be the clauses in the path decomposition order as explained in the beginning of this sub-section. We will first show that places representing clauses can be added to the bags of the above decomposition without increasing their size much, while maintaining the invariant that all bags containing any one place are connected in the decomposition. We will do this by **augmenting** existing bags with new elements: if B is any bag in the decomposition and p is an element not in B , augmenting B with p means creating a new bag B' immediately to the left of B containing p in addition to the elements in B . We will call the new bag B' thus created an augmented bag. Perform the following operation in increasing order for each j between 1 and m : if B is the first non-augmented bag from left to contain all literals of the clause C_j , augment B with C_j .

There will be m new bags created due to the above augmentation steps. Due to the path decomposition ordering of C_1, \dots, C_m , the augmented bag containing C_{j+1} occurs to the right of the augmented bag containing C_j for each j , $1 \leq j < m$. There might be some non-augmented bags between the augmented bags containing C_j and C_{j+1} . If so, add C_j to such non-augmented bags. Now, to every bag, if it contains C_j for some j between 1 and m , add C_{j+1} . It is routine to verify the following properties of the sequence we have with the bags modified as above.

- Each bag has at most $3p_w + 4k + 8$ elements.
- The set of bags containing any one element forms a contiguous sub-sequence.
- Every vertex and edge in any subgraph induced by the parts of the net in Fig. 1, Fig. 3, and Fig. 4 is contained in some bag.

To account for the subgraph induced by the parts of the net in Fig. 2, we append the following sequence of bags for each r between 1 and k :

$$\begin{aligned} & (\{tu_r^0, tu_r^1\} \cup P_1) - (\{tu_r^1, tu_r^2\} \cup P_1) - \dots - (\{tu_r^{tg(r)-1}, tu_r^{tg(r)}\} \cup P_1) - \\ & (\{fl_r^0, fl_r^1\} \cup P_1) - (\{fl_r^1, fl_r^2\} \cup P_1) - \dots - (\{fl_r^{n[r]-tg(r)-1}, fl_r^{n[r]-tg(r)}\} \cup P_1) \end{aligned}$$

The resulting sequence of bags is a path decomposition of the flow graph of the Petri net, whose width is at most $3p_w + 4k + 7$. \square

In the above reduction, it is enough to check if in the constructed 1-safe net, we can reach a marking that has a token at the place g . This can be expressed as reachability, coverability etc. Hence we get:

Theorem 8. *With the pathwidth (and hence treewidth also) of the flow graph of a 1-safe Petri net as parameter, reachability, coverability, CTL model checking and the complement of LTL/MSO model checking (with formulas of constant size) are $W[1]$ -hard.*

3.2 Graph pebbling problems, treewidth and pathwidth

The techniques used in the above lower bound proof can be easily translated to some graph pebbling problems [6]. As conjectured in [6, section 5], we prove that SIGNED DIGRAPH PEBBLING I, parameterized by treewidth is $W[1]$ -hard. An instance of this problem has a bipartite digraph $D = (V, A)$ for which the vertex set V is partitioned $V = Red \cup Blue$, and also the arc set A is partitioned into two partitions $A = A^+ \cup A^-$. The problem is to reach the *finish state* where there are pebbles on all the red vertices, starting from a *start state* where there are no pebbles on any of the red vertices, by a series of moves of the following form:

- If b is a blue vertex such that for all s such that $(s, b) \in A^+$, s is pebbled, and for all s such that $(s, b) \in A^-$, s is not pebbled (in which case we say that b is *enabled*), then the set of vertices s such that $(b, s) \in A^+$ are reset by making them all pebbled, and the set of all vertices s such that $(b, s) \in A^-$ are reset by making them all unpebbled.

Corollary 9. *Parameterized by pathwidth (and hence by treewidth also), SIGNED DIGRAPH PEBBLING is $W[1]$ -hard.*

Proof. To reduce p-PW-SAT to SIGNED DIGRAPH PEBBLING, we first reduce the given p-PW-SAT instance to a 1-safe net as shown in Lemma 6. From this 1-safe net, construct an instance of SIGNED DIGRAPH PEBBLING as follows. Let the set of all places form the set of vertices *Red* and the set of all transitions form the set of vertices *Blue*. The arcs of the SIGNED DIGRAPH PEBBLING instance are as follows.

1. If $Pre(p, t) = 1$ in the 1-safe net, draw an A^+ arc from p to t in the SIGNED DIGRAPH PEBBLING instance.

2. If $Pre(p, t) = 1$ and $Post(p, t) = 0$, draw an A^- arc from t to p .
3. If $Pre(p, t) = 0$ and $Post(p, t) = 1$, draw an A^+ arc from t to p .

Suppose that in the 1-safe net, $M_1 \xRightarrow{t} M_2$. It is clear that the constructed SIGNED DIGRAPH PEBBLING instance in the state where precisely those red vertices are pebbled that have a token in M_1 enables the blue vertex t , and can move to the state where precisely those red vertices are pebbled that have a token in M_2 . Add a special blue vertex b_1 with A^+ arcs from b_1 to $q_1, q_2, \dots, q_n, tu_1^0, \dots, tu_k^0, fl_1^0, \dots, fl_k^0, C_1$ and s . Add A^- arcs from all red vertices to b_1 . In the start state where there no pebbles at all, b_1 is the only blue vertex enabled. The blue vertex b_1 is enabled only in the start state. Upon performing the legal move using b_1 from the start state, we will reach a state in which precisely those red vertices are pebbled that have a token in the initial marking of the 1-safe net. From this state, there is at least one pebbled red vertex in any reachable state, so b_1 is never enabled again. From this state, we can reach a state with the red vertex g pebbled iff the given p-PW-SAT instance is a YES instance. Add another special blue vertex b_2 with an A^+ arc from the red vertex g to b_2 . Add A^+ arcs from b_2 to all red vertices. All blue vertices except b_1 and b_2 unpebble at least one red vertex. Hence, the only way to reach the finish state (where all red vertices must be pebbled) from the start state is to enable b_2 . The only way to enable b_2 is to reach a state where the red vertex g is pebbled. Hence, the constructed SIGNED DIGRAPH PEBBLING instance is a YES instance iff the given p-PW-SAT instance is a YES instance.

To complete the reduction, it only remains to show that the pathwidth of the SIGNED DIGRAPH PEBBLING instance is bounded by the pathwidth of the flow graph of the intermediate 1-safe net. Consider an optimal path decomposition of this flow graph. For every transition t , the set of all input and output places of t forms a clique in the flow graph. Hence, there will be at least one bag B in the path decomposition containing all these places. Create an extra bag B' adjacent to B containing all elements of B and also the blue vertex corresponding to t . After doing this for each transition, add the vertices b_1 and b_2 to all bags. The resulting decomposition is a path decomposition of the SIGNED DIGRAPH PEBBLING instance. Its width is at most 3 more than the pathwidth of the flow graph of the 1-safe net. \square

3.3 1-safe Petri nets and benefit depth

Here we show that the parameter benefit depth is not helpful for 1-safe Petri nets, by showing W[1]-hardness using a parameterized reduction from the constraint satisfaction problem (CSP).

Theorem 10. *With benefit depth as the parameter in 1-safe Petri nets, reachability, coverability, CTL model checking and the complement of the LTL/MSO model checking problems, even with formulas of constant size, are W[1]-hard.*

The rest of this section is devoted to a proof of the above theorem. To show that with benefit depth as parameter, reachability in 1-safe nets is W[1]-hard, we

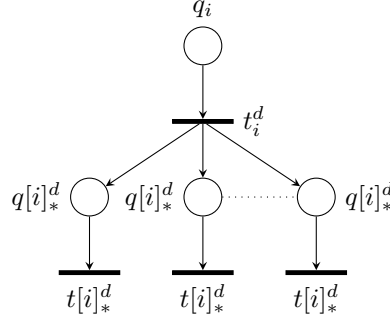


Fig. 5. Part of the net for every variable q_i and domain value d

will show a FPT reduction from the constraint satisfaction problem (CSP). With the size of the domain dom and the maximum number of constraints in which any one variable can occur (called degree) deg as parameters, CSP is $W[1]$ -hard [21, Corollary 2]. Given an instance of CSP with domain size dom , degree deg , n variables and m constraints, we construct a 1-safe net with the following places.

1. For every variable q_i , a place q_i .
2. For every constraint C_j where j is between 1 and m , a place C_j .
3. For every i between 1 and n , for every domain element d between 1 and dom , for every constraint C_j in which q_i appears, the place $q[i]_j^d$.
4. One place g for checking that all constraints are satisfied.

We assume without loss of generality that every variable occurs in at least one constraint. Construction of the 1-safe net is illustrated in the following diagrams. For every variable q_i and domain value d (between 1 and dom), part of the net shown in Fig. 5 is constructed. Intuitively, the transition t_i^d is fired to assign domain value d to q_i . In Fig. 5, the set of places labelled by $q[i]_*^d$ should be understood to stand for the set of places $\{q[i]_j^d \mid q_i \text{ occurs in constraint } C_j\}$. The set of transitions labelled $t[i]_*^d$ should be similarly understood.

For every constraint C_j and every admissible tuple of domain values for C_j , part of the net shown in Fig. 6 is constructed. In Fig. 6, it is assumed that the constraint C_j consists of variables q_1, q_2 and q_3 and that $(3, 5, 6)$ is an admissible tuple for this constraint. Finally, the part of the net in Fig. 7 verifies that all constraints are satisfied. The initial marking has 1 token each in each of the places q_1, \dots, q_n and 0 tokens in all other places. The final marking to be reached is 1 token at the place g and 0 tokens in all other places.

Lemma 11. *Given a CSP instance of domain size dom and degree deg , the benefit depth of the 1-safe net constructed above is at most $2 + deg(dom + 1)$. The given CSP instance is satisfiable iff the required final marking is reachable from the initial marking in the constructed 1-safe net.*

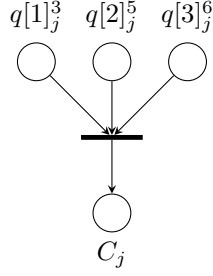


Fig. 6. Part of the net for every constraint C_j and every admissible tuple

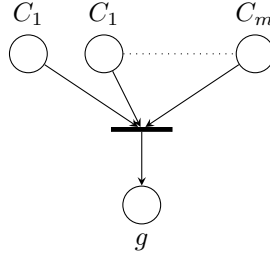


Fig. 7. Part of the net to check that all constraints are satisfied

Proof. Maximum number of places are benefited by some place in $\{q_1, \dots, q_n\}$. Any place q_i can benefit itself, the place g , the set of places $\{q[i]_j^d \mid 1 \leq d \leq \text{dom}, q_i \text{ occurs in } C_j\}$ and at most deg places among $\{C_1, \dots, C_m\}$. This adds up to at most $2 + \text{deg}(\text{dom} + 1)$.

Suppose the given CSP instance is satisfiable. For each variable q_i , if d is the domain value assigned to q_i by the satisfying assignment, fire the transition t_i^d shown in Fig. 5. Since the satisfying assignment satisfies all the constraints, the transitions shown in Fig. 6 can be fired to get a token into each of the places C_1, \dots, C_m . Then the transition shown in Fig. 7 can be fired to get a token in the place g . Any tokens remaining in places $q[i]_*^d$ can be removed by firing transitions $t[i]_*^d$ shown in Fig. 5. Now, the token in the place g is the only token in the entire net and this is the final marking required to be reached.

Suppose the required final marking is reachable in the constructed 1-safe net. Consider any firing sequence reaching the required final marking. Since the final marking needs a token in the place g and the only transition that can add token to g is the one shown in Fig. 7, the firing sequence fires this transition. For this transition to be enabled, a token needs to be present in each of the places C_1, \dots, C_m . These tokens can only be added by firing transitions shown in Fig. 6. To fire these transitions, tokens need to be present in the places $q[i]_*^d$. To generate these tokens, the firing sequence would have to fire some transition

t_i^d for each i between 1 and n . Consider the assignment that assigns domain value d to q_i iff the firing sequence fired t_i^d . By construction, this assignment satisfies all constraints. \square

Since the 1-safe net described above can be constructed in time polynomial in the size of the given CSP instance, Lemma 11 shows that this reduction is a parameterized reduction from CSP (with *dom* and *deg* as parameters) to reachability in 1-safe nets (with benefit depth as the parameter). In the above reduction, it is enough to check if in the constructed 1-safe net, we can reach a marking that has a token at the place g . This can be expressed as reachability, coverability etc. This proves Theorem 10.

4 Vertex cover and model checking 1-safe Petri nets

In this section, we will show that with the vertex cover number of the flow graph of the given 1-safe Petri net and the size of the given LTL/MSO formula as parameters, checking whether the given net is a model of the given formula is FPT. With vertex cover number as the only parameter, we cannot hope to get this kind of tractability:

Proposition 12. *Model checking LTL (and hence MSO) formulas on 1-safe Petri nets whose flow graph has constant vertex cover number is CO-NP-hard.*

Proof. We give a reduction from the complement of propositional logic satisfiability problem. Let \mathcal{F} be a propositional formula over variables q_1, \dots, q_n . Consider the 1-safe net $\mathcal{N}_{\mathcal{F}}$ shown in Fig. 8. The initial marking consists of 0

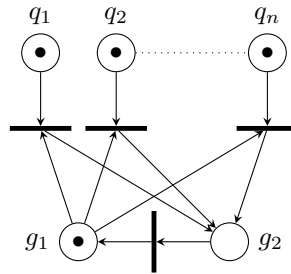


Fig. 8. The net $\mathcal{N}_{\mathcal{F}}$ associated with a propositional formula \mathcal{F}

tokens in g_2 and 1 token each in all other places. The flow graph of $\mathcal{N}_{\mathcal{F}}$ has a vertex cover of size 2 ($\{g_1, g_2\}$). Every marking M reachable in $\mathcal{N}_{\mathcal{F}}$ defines an assignment to the variables used in \mathcal{F} : $q_i = \top$ iff $M(q_i) = 1$. Every assignment can be represented by some reachable marking in this way. We claim that \mathcal{F} is not satisfiable iff $\mathcal{N}_{\mathcal{F}}$ is a model of the LTL formula $\neg(\top \text{ Until } \mathcal{F})$. If \mathcal{F} is not

satisfiable, then none of the markings reachable in $\mathcal{N}_{\mathcal{F}}$ satisfies \mathcal{F} . Hence, $\mathcal{N}_{\mathcal{F}}$ is a model of the LTL formula $\neg(\top \text{ Until } \mathcal{F})$. On the other hand, if $\mathcal{N}_{\mathcal{F}}$ is a model of $\neg(\top \text{ Until } \mathcal{F})$, then none of the markings reachable in $\mathcal{N}_{\mathcal{F}}$ satisfies \mathcal{F} . Hence, \mathcal{F} is not satisfiable. \square

Since a run of a 1-safe net \mathcal{N} with set of places P is a sequence of subsets of P , we can think of such sequences as strings over the alphabet $\mathcal{P}(P)$ (the power set of P). It is known [2, 22] that with any LTL or MSO formula ϕ , we can associate a finite state automaton \mathcal{A}_{ϕ} over the alphabet $\mathcal{P}(P)$ accepting the set of finite strings which are its models, as well as a finite state Büchi automaton \mathcal{B}_{ϕ} accepting the set of infinite string models.

Figure 9 shows the schematic of a simple manufacturing system modelled as a 1-safe Petri net. Starting from p_1 , it picks up one unit of a raw material α and goes to p_2 , then picks up raw material β , then γ . Transition t_1 does some processing and then the system starts from p_1 again. Suppose we want to make sure that whenever the system picks up a unit of raw material β , it is processed immediately. In other words, whenever the system stops at a marking where no transitions are enabled, there should not be a token in p_3 . This can be checked by verifying that all finite maximal runs satisfy the formula $\forall x((\forall y \ y \leq x) \Rightarrow \neg p_3(x))$. The satisfaction of this formula depends only on the number of units of raw materials α, β and γ at the beginning, i.e., the number of tokens at the initial marking. The naive approach of constructing the whole reachability graph results in an exponentially large state space, due to the different orders in which the raw materials of each type can be drawn. If we want to reason about only the central system (which is the vertex cover $\{p_1, p_2, p_3, p_4\}$ in the above system), it turns out that we can ignore the order and express the requirements on the numbers by integer linear constraints.

Suppose VC is a vertex cover for $G(\mathcal{N})$. We use the fact that if $v_1, v_2 \notin VC$ are two vertices not in VC that have the same set of neighbours, v_1 and v_2 have similar properties. This has been used to obtain FPT algorithms for many hard problems (e.g. [9]). The following definitions formalize this.

Definition 13. *Let VC be a vertex cover of $G(\mathcal{N})$. The **(VC-) neighbourhood** of a transition t is the ordered pair $(\bullet t \cap VC, t \bullet \cap VC)$. We denote by l the number of different VC-neighbourhoods.*

Definition 14. *Suppose \mathcal{N} is a Petri net with l neighbourhoods for vertex cover VC , and $p \notin VC$. The **(VC-) interface** $\text{int}[p]$ of p is defined as the function $\text{int}[p] : \{1, \dots, l\} \rightarrow \mathcal{P}(\{-1, 1\})$, where for every j between 1 and l and every $w \in \{1, -1\}$, there is a transition t_j of VC-neighbourhood j such that $w = -\text{Pre}(p, t_j) + \text{Post}(p, t_j)$ iff $w \in \text{int}[p](j)$.*

In the net in Fig. 9 with $VC = \{p_1, p_2, p_3, p_4\}$, all transitions labelled α have the same VC-neighbourhood and all the corresponding places have the same VC-interface. Since there can be $2k$ arcs between a transition and places in VC if $|VC| = k$, there can be at most 2^{2k} different VC-neighbourhoods of transitions. There are at most $4^{2^{2k}}$ VC-interfaces. The set of interfaces is denoted by Int .

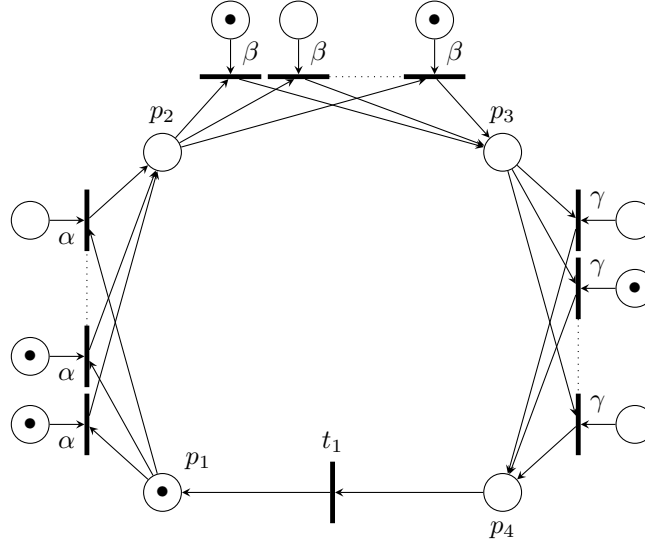


Fig. 9. An example of a system with small vertex cover

Proposition 15. *Let \mathcal{N} be a 1-safe net with VC being a vertex cover of $G(\mathcal{N})$. Let p_1, p_2, \dots, p_i be places not in the vertex cover, all with the same interface. Let M be some marking reachable from the initial marking of \mathcal{N} . If $M(p_j) = 1$ for some j between 1 and i , then M does not enable any transition that adds tokens to any of the places p_1, \dots, p_i .*

Proof. Suppose there is a transition t enabled at M that adds a token to $p_{j'}$ for some j' between 1 and i . Then there is a transition t' with the same neighbourhood as t (and hence enabled at M too) that can add a token to p_j . Firing t' from M will create 2 tokens at p_j , contradicting the fact that \mathcal{N} is 1-safe. \square

If the initial marking has tokens in many places with the same interface, then no transition can add tokens to any of those places until all the tokens in all those places are removed. Once all tokens are removed, one of the places can receive one token after which, no place can receive tokens until this one is removed. All these places have the same interface. Thus, a set of places with the same interface can be thought of as an initial storehouse of tokens, after depleting which it can be thought of as a single place. However, a formula in our logic can reason about individual places, so we still need to keep track of individual places that occur in the formula.

Proposition 16. *Let \mathcal{N} be a 1-safe net and ϕ be an MSO formula. Let $P_\phi \subseteq P$ be the subset of places that occur in ϕ . Let $\pi = M_0 M_1 \dots$ and $\pi' = M'_0 M'_1 \dots$ be two finite or infinite runs of \mathcal{N} such that for all positions j of π and for all $p \in P_\phi$, $M_j(p) = M'_j(p)$. For any assignment s , we have $\pi, s \models \phi$ iff $\pi', s \models \phi$.*

Proof. By a straightforward induction on the structure of ϕ . \square

Let \mathcal{N} be a 1-safe net such that $G(\mathcal{N})$ has a vertex cover VC of size k . Suppose ϕ is a formula and we have to check if \mathcal{N} satisfies ϕ . For each interface I , let $P_I \subseteq P$ be the places not in VC with interface I . If $P_I \setminus P_\phi \neq \emptyset$ (i.e., if there are places in P_I that are not in ϕ), designate one of the places in $P_I \setminus P_\phi$ as p_I . Define the set of special places $\mathcal{S} = VC \cup P_\phi \cup \{p_I \in P_I \setminus P_\phi \mid I \text{ is an interface and } P_I \setminus P_\phi \neq \emptyset\}$. Note that $|\mathcal{S}| \leq k + |\phi| + 4^{2^k}$. Since this number is a function of the parameters of the input instance, we will treat it as a parameter.

We need a structure that keeps track of changes in places belonging to \mathcal{S} , avoiding a construction involving all reachable markings. This can be done by a finite state machine whose states are subsets of \mathcal{S} . Transitions of the Petri net that only affect places in \mathcal{S} can be simulated by the finite state machine with its usual transitions. To simulate transitions of the net that affect places outside \mathcal{S} , we need to impose some conditions on the number of times transitions of the finite state machine can be used. The following definition formalizes this. For a marking M of \mathcal{N} , let $M[\mathcal{S} = \{p \in \mathcal{S} \mid M(p) = 1\}]$.

Definition 17. Given a 1-safe net \mathcal{N} with initial marking M_0 and \mathcal{S} defined from ϕ as above, the **edge constrained automaton** $\mathcal{A}_{\mathcal{N}} = (Q_{\mathcal{N}}, \Sigma, \delta_{\mathcal{N}}, u, F_{\mathcal{N}})$ is a structure defined as follows. $Q_{\mathcal{N}} = \mathcal{P}(\mathcal{S})$ and $\Sigma = \text{Int} \cup \{\perp\}$ (recall that Int is the set of interfaces in \mathcal{N}). The transition relation $\delta \subseteq Q_{\mathcal{N}} \times \Sigma \times Q_{\mathcal{N}}$ is such that for all $P_1, P_2 \subseteq \mathcal{S}$ and $I \in \text{Int} \cup \{\perp\}$, $(P_1, I, P_2) \in \delta$ iff there are markings M_1, M_2 and a transition t of \mathcal{N} such that

- $M_1[\mathcal{S} = P_1, M_2[\mathcal{S} = P_2$ and $M_1 \xrightarrow{t} M_2$,
- t removes a token from a place $p \in P_I \setminus \mathcal{S}$ of interface I if $I \in \text{Int}$ and
- t does not have any of its input or output places in $P \setminus \mathcal{S}$ if $I = \perp$.

The **edge constraint** $u : \text{Int} \rightarrow \mathbb{N}$ is given by $u(I) = |\{p \in P_I \setminus \mathcal{S} \mid M_0(p) = 1\}|$. A subset $P_1 \subseteq \mathcal{S}$ is in $F_{\mathcal{N}}$ iff for every marking M with $M[\mathcal{S} = P_1$, the only transitions enabled at M remove tokens from some place not in \mathcal{S} .

Intuitively, the edge constraint u defines an upper bound on the number of times those transitions can be used that reduce tokens from places not in \mathcal{S} .

Definition 18. Let $\mathcal{A}_{\mathcal{N}}$ be an edge constrained automaton as in Def. 17 and let $\pi = P_0 P_1 \dots$ be a finite or infinite word over $\mathcal{P}(\mathcal{S})$. Then π is a valid run of $\mathcal{A}_{\mathcal{N}}$ iff for every position $j \geq 1$ of π , we can associate an element $I_j \in \Sigma$ such that

- for every position $j \geq 1$ of π , $(P_{j-1}, I_j, P_j) \in \delta$ and
- for every $I \in \text{Int}$, $|\{j \geq 1 \mid I_j = I\}| \leq u(I)$.
- if π is finite and P_j is the last element of π , then $P_j \in F_{\mathcal{N}}$ and for every interface $I \in \text{Int}$ and marking $M_j[\mathcal{S} = P_j$ enabling some transition that removes tokens from some place in $P_I \setminus \mathcal{S}$, $|\{j \geq 1 \mid I_j = I\}| = u(I)$.

Next we have a run construction lemma.

Lemma 19. *Let \mathcal{N} be a 1-safe net with initial marking M_0 , ϕ be a formula and $\mathcal{A}_{\mathcal{N}}$ be as in Def. 17. For every infinite (maximal finite) run $\pi = M_0M_1 \cdots$ of \mathcal{N} , there exists an infinite (finite) run $\pi' = M'_0M'_1 \cdots$ such that the word $(M'_0[\mathcal{S}](M'_1[\mathcal{S}]) \cdots$ is a valid run of $\mathcal{A}_{\mathcal{N}}$ and for every position j of π , $M'_j[P_\phi] = M_j[P_\phi]$. If an infinite (finite) word $\pi = P_0P_1 \cdots$ over $\mathcal{P}(\mathcal{S})$ is a valid run of $\mathcal{A}_{\mathcal{N}}$ and $P_0 = M_0[\mathcal{S}]$, then there is an infinite (finite maximal) run $M_0M_1 \cdots$ of \mathcal{N} such that $M_j[\mathcal{S}] = P_j$ for all positions j of π .*

Proof. Let $\pi = M_0M_1 \cdots$ be an infinite or a maximal finite run of \mathcal{N} . For every interface $I \in \text{Int}$, perform the following steps: if for some marking M in the above run, $\{p \in P_I \mid M(p) = 1\} = \emptyset$, let M_I be the first such marking. By Prop. 15, no transition occurring before M_I will add any token to any place in P_I . If there is any transition occurring after M_I that adds/removes tokens from $P_I \setminus \mathcal{S}$, replace it with another transition with the same neighbourhood that adds/removes tokens from p_I . By Prop. 15, such a replacement will not affect any place in P_ϕ and the new sequence of transitions is still enabled at M_0 . After performing this process for every interface $I \in \text{Int}$, let the new run be $\pi' = M'_0M'_1 \cdots$. By construction, we have $M'_j[P_\phi] = M_j[P_\phi]$ for all positions $j \geq 0$ of π . If π is a maximal finite run, so is π' .

Now we will prove that the word $(M'_0[\mathcal{S}](M'_1[\mathcal{S}]) \cdots$ is a valid run of $\mathcal{A}_{\mathcal{N}}$. Suppose the sequence of transitions producing the run π' is $M'_0 \xrightarrow{t_1} M'_1 \xrightarrow{t_2} M'_2 \cdots$. For each position $j \geq 1$ of this run, define $I_j \in \Sigma$ as follows:

- if t_j has all its input and output places among places \mathcal{S} , let $I_j = \perp$.
- if t_j removes a token from some place in $P_I \setminus \mathcal{S}$ for some interface I , let $I_j = I$. Due to the way π' is constructed, this kind of transition can only occur before the position of M_I and the number of such occurrences is at most $|\{p \in P_I \setminus \mathcal{S} \mid M_0(p) = 1\}| = u(I)$.

Due to the way π' is constructed, there will not be any transition that adds tokens to any place in $P_I \setminus \mathcal{S}$ for any interface I . By definition, it is clear that for every position $j \geq 1$ of π' , $(M'_{j-1}[\mathcal{S}], I_j, M'_j[\mathcal{S}]) \in \delta_{\mathcal{N}}$. In addition, for every interface $I \in \text{Int}$, we have $|\{j \geq 1 \mid I_j = I\}| \leq u(I)$. Hence, the word $(M'_0[\mathcal{S}](M'_1[\mathcal{S}]) \cdots$ is a valid run of $\mathcal{A}_{\mathcal{N}}$ if the word is infinite. If π' is finite, suppose M'_r is the last marking of the sequence π' . Suppose for some variety $I \in \text{Int}$, there is some marking M such that $M[\mathcal{S}] = M'_r[\mathcal{S}]$ and M enables some transition t that removes tokens from some place in $P_I \setminus \mathcal{S}$. Since M'_r does not enable any transition, all transitions (including t) removing tokens from some place in $P_I \setminus \mathcal{S}$ are disabled in M'_r . This means that every place in $P_I \setminus \mathcal{S}$ that had a token in M_0 has lost its token in M'_r . Since such loss of tokens can only happen by firing transitions that remove tokens from places in $P_I \setminus \mathcal{S}$, we have $|\{j \geq 1 \mid I_j = I\}| = u(I)$. Hence, to prove that $(M'_0[\mathcal{S}](M'_1[\mathcal{S}]) \cdots (M'_r[\mathcal{S}])$ is a valid run of $\mathcal{A}_{\mathcal{N}}$, it is left to show that $M'_r \in F_{\mathcal{N}}$. To see that this is true, observe that if some marking M with $M[\mathcal{S}] = M'_r$ enables a transition that does not remove any token from $P \setminus \mathcal{S}$, then so does M'_r , a contradiction.

Next, suppose $\pi = P_0P_1 \cdots$ is an infinite or finite word that is a valid run of $\mathcal{A}_{\mathcal{N}}$ such that $P_0 = M_0[\mathcal{S}]$. For every position $j \geq 1$ of π , there are $I_j \in$

$Int \cup \{\perp\}$, transition t'_j and markings M'_{j-1} and M'_j such that $(P_{j-1}, I_j, P_j) \in \delta_{\mathcal{N}}$, $M'_{j-1} \xrightarrow{t'_j} M'_j$, $M'_{j-1} \upharpoonright \mathcal{S} = P_{j-1}$ and $M'_j \upharpoonright \mathcal{S} = P_j$. Define transitions t_i as follows:

- If $I_j = \perp$, transition t'_j has all its input and output places in \mathcal{S} . Let $t_j = t'_j$.
- If $I_j = I \in Int$, transition t'_j removes a token from some place in $P_I \setminus \mathcal{S}$. Let t' be a transition of the same neighbourhood as t'_j that removes a token from some place $p_j \in \{p \in P_I \setminus \mathcal{S} \mid M_0(p) = 1\}$ such that no transition among t_1, \dots, t_{j-1} removes tokens from p_j . This is possible since, due to the validity of π in $\mathcal{A}_{\mathcal{N}}$, $|\{j' \geq 1 \mid I_{j'} = I\}| \leq |\{p \in P_I \setminus \mathcal{S} \mid M_0(p) = 1\}| = u(I)$. Let $t_j = t'$.

We will now prove by induction on j that there are markings M_0, M_1, \dots such that $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_j} M_j$ and $M_j \upharpoonright \mathcal{S} = P_j$ for every position j of π .

Base case $j = 1$: If $I_1 = \perp$, the fact that $M_0 \upharpoonright \mathcal{S} = P_0$, $M'_0 \xrightarrow{t_1} M'_1$ and that t_1 has all its input and output places in \mathcal{S} implies that $M_0 \xrightarrow{t_1} M_1$ for some M_1 such that $M_1 \upharpoonright \mathcal{S} = P_1$. If $I_j = I \in Int$, then t_1 removes a token from some place $p_1 \in P_I \setminus \mathcal{S}$. Again the fact that $M_0 \upharpoonright \mathcal{S} = P_0$ and $M'_0 \xrightarrow{t'_1} M'_1$ implies that $M_0 \xrightarrow{t_1} M_1$ for some M_1 such that $M_1 \upharpoonright \mathcal{S} = P_1$.

Induction step: If $I_{j+1} = \perp$, the fact that $M_j \upharpoonright \mathcal{S} = P_j$, $M'_j \xrightarrow{t_{j+1}} M'_{j+1}$ and that t_{j+1} has all its input and output places in \mathcal{S} implies that $M_j \xrightarrow{t_{j+1}} M_{j+1}$ for some M_{j+1} such that $M_{j+1} \upharpoonright \mathcal{S} = P_{j+1}$. If $I_{j+1} = I \in Int$, then t_{j+1} removes a token from some place $p_{j+1} \in P_I \setminus \mathcal{S}$. Again the fact that $M_j \upharpoonright \mathcal{S} = P_j$ and $M'_j \xrightarrow{t'_{j+1}} M'_{j+1}$ implies that $M_j \xrightarrow{t_{j+1}} M_{j+1}$ for some M_{j+1} such that $M_{j+1} \upharpoonright \mathcal{S} = P_{j+1}$.

If π is a finite word, we have to prove that the run constructed above is a maximal finite run. Let M_r be the last marking in the sequence constructed above. We will prove that M_r does not enable any transition. Suppose some transition t is enabled at M_r . Since $M_r \upharpoonright \mathcal{S} \in F_{\mathcal{N}}$, t removes a token from some place in $P_I \setminus \mathcal{S}$ for some variety I . Since $|\{j \geq 1 \mid I_j = I\}| = u(I)$, there are $u(I)$ transition occurrences among t_1, \dots, t_r that each remove a token from some place in $P_I \setminus \mathcal{S}$. Since there were exactly $u(I)$ places in $P_I \setminus \mathcal{S}$ that had a token in M_0 and no other transition adds any token to any place in $P_I \setminus \mathcal{S}$, t can not be enabled at M_r . Hence, no transition is enabled at M_r . \square

Lemma 19 implies that in order to check if \mathcal{N} is a model of the formula ϕ , it is enough to check that all valid runs of $\mathcal{A}_{\mathcal{N}}$ satisfy ϕ . This can be done by checking that no finite valid run of $\mathcal{A}_{\mathcal{N}}$ is accepted by $\mathcal{A}_{\neg\phi}$ and no infinite valid run of $\mathcal{A}_{\mathcal{N}}$ is accepted by $\mathcal{B}_{\neg\phi}$. As usual, this needs a product construction. Automata $\mathcal{A}_{\neg\phi}$ and $\mathcal{B}_{\neg\phi}$ run on the alphabet $\mathcal{P}(P_{\phi})$. Let $Q_{\mathcal{A}}$ and $Q_{\mathcal{B}}$ be the set of states of $\mathcal{A}_{\neg\phi}$ and $\mathcal{B}_{\neg\phi}$ respectively. Then, $\mathcal{A}_{\neg\phi} = (Q_{\mathcal{A}}, \mathcal{P}(P_{\phi}), \delta_{\mathcal{A}}, Q_{0\mathcal{A}}, F_{\mathcal{A}})$ and $\mathcal{B}_{\neg\phi} = (Q_{\mathcal{B}}, \mathcal{P}(P_{\phi}), \delta_{\mathcal{B}}, Q_{0\mathcal{B}}, F_{\mathcal{B}})$.

Definition 20. $\mathcal{A}_{\mathcal{N}} \times \mathcal{A}_{\neg\phi} = (Q_{\mathcal{N}} \times Q_{\mathcal{A}}, \Sigma, \delta_{\mathcal{A}}^{\mathcal{N}}, \{M_0[\mathcal{S}] \times Q_{0\mathcal{A}}, F_{\mathcal{N}} \times F_{\mathcal{A}}, u\})$,
 $\mathcal{A}_{\mathcal{N}} \times \mathcal{B}_{\neg\phi} = (Q_{\mathcal{N}} \times Q_{\mathcal{B}}, \Sigma, \delta_{\mathcal{B}}^{\mathcal{N}}, \{M_0[\mathcal{S}] \times Q_{0\mathcal{B}}, Q_{\mathcal{N}} \times F_{\mathcal{B}}, u\})$ where

$$\begin{aligned} ((q_1, q_2), I, (q'_1, q'_2)) &\in \delta_{\mathcal{A}}^{\mathcal{N}} \text{ iff } (q_1, I, q'_1) \in \delta_{\mathcal{N}} \text{ and } (q_2, q_1 \cap P_{\phi}, q'_2) \in \delta_{\mathcal{A}} \\ ((q_1, q_2), I, (q'_1, q'_2)) &\in \delta_{\mathcal{B}}^{\mathcal{N}} \text{ iff } (q_1, I, q'_1) \in \delta_{\mathcal{N}} \text{ and } (q_2, q_1 \cap P_{\phi}, q'_2) \in \delta_{\mathcal{B}} \end{aligned}$$

An accepting path of $\mathcal{A}_{\mathcal{N}} \times \mathcal{A}_{\neg\phi}$ is a sequence $(q_0, q'_0)I_1(q_1, q'_1) \cdots I_r(q_r, q'_r)$ which is $\delta_{\mathcal{A}}^{\mathcal{N}}$ -respecting:

- $(q_0, q'_0), (q_1, q'_1), \dots, (q_r, q'_r) \in Q_{\mathcal{N}} \times Q_{\mathcal{A}}$,
- the word $I_1 \cdots I_r \in \Sigma^*$ witnesses the validity of the run $q_0q_1 \cdots q_r$ in $\mathcal{A}_{\mathcal{N}}$ (as in Def. 18) and
- the word $(q_0 \cap P_{\phi}) \cdots (q_r \cap P_{\phi})$ is accepted by $\mathcal{A}_{\neg\phi}$ through the run $q'_0q'_1 \cdots q'_rq'_F$ for some $q'_F \in F_{\mathcal{A}}$ with $(q'_r, q_r \cap P_{\phi}, q'_F) \in \delta_{\mathcal{A}}$.

An accepting path of $\mathcal{A}_{\mathcal{N}} \times \mathcal{B}_{\neg\phi}$ is defined similarly.

Proposition 21. A 1-safe net \mathcal{N} with initial marking M_0 is a model of a formula ϕ iff there is no accepting path in $\mathcal{A}_{\mathcal{N}} \times \mathcal{A}_{\neg\phi}$ and $\mathcal{A}_{\mathcal{N}} \times \mathcal{B}_{\neg\phi}$.

Proof. Suppose \mathcal{N} is a model of ϕ . Hence, all maximal runs of \mathcal{N} satisfy ϕ . We will prove that there is no accepting path in $\mathcal{A}_{\mathcal{N}} \times \mathcal{A}_{\neg\phi}$ and $\mathcal{A}_{\mathcal{N}} \times \mathcal{B}_{\neg\phi}$. Assume by way of contradiction that there is an accepting path $(q_0, q'_0)I_1(q_1, q'_1) \cdots I_r(q_r, q'_r)$ in $\mathcal{A}_{\mathcal{N}} \times \mathcal{A}_{\neg\phi}$. By Def. 20, $q_0q_1 \cdots q_r$ is a valid run of $\mathcal{A}_{\mathcal{N}}$. By Lemma 19, there is a finite maximal run $M_0M_1 \cdots M_r$ of \mathcal{N} with $M_j[\mathcal{S}] = q_j$ for all positions $0 \leq j \leq r$. By Def. 20, $(q_0 \cap P_{\phi}) \cdots (q_r \cap P_{\phi})$ is accepted by $\mathcal{A}_{\neg\phi}$ and hence satisfies $\neg\phi$. Proposition 16 now implies that $M_0M_1 \cdots M_r$ satisfies $\neg\phi$, a contradiction. The argument for $\mathcal{A}_{\mathcal{N}} \times \mathcal{B}_{\neg\phi}$ is similar.

Suppose \mathcal{N} is not a model of ϕ . Suppose there is a finite maximal run $M_0M_1 \cdots M_r$ of \mathcal{N} that satisfies $\neg\phi$. By Lemma 19, there is a finite maximal run $\pi' = M'_0M'_1 \cdots M'_r$ such that the word $(M'_0[\mathcal{S}](M'_1[\mathcal{S}]) \cdots (M'_r[\mathcal{S}])$ is a valid run of $\mathcal{A}_{\mathcal{N}}$ and for every position j of π' , $M'_j[P_{\phi}] = M_j[P_{\phi}]$. By Prop. 16, $(M'_0[P_{\phi}](M'_1[P_{\phi}]) \cdots (M'_r[P_{\phi}])$ satisfies $\neg\phi$ and hence accepted by $\mathcal{A}_{\neg\phi}$, say with the run $q'_0q'_1 \cdots q'_rq'_F$. Let the word $I_1 \cdots I_r \in \Sigma^*$ witness the validity of the run $(M'_0[\mathcal{S}](M'_1[\mathcal{S}]) \cdots (M'_r[\mathcal{S}])$ in $\mathcal{A}_{\mathcal{N}}$, as in Def. 18. By Def. 20, the sequence $(M'_0[P_{\phi}, q'_0]I_1(M'_1[P_{\phi}, q'_1]) \cdots I_r(M'_r[P_{\phi}, q'_r])$ is an accepting path of $\mathcal{A}_{\mathcal{N}} \times \mathcal{A}_{\neg\phi}$. The argument for maximal infinite runs is similar. \square

To efficiently check the existence of accepting paths in $\mathcal{A}_{\mathcal{N}} \times \mathcal{A}_{\neg\phi}$ and $\mathcal{A}_{\mathcal{N}} \times \mathcal{B}_{\neg\phi}$, it is convenient to look at them as graphs, possibly with self loops and parallel edges. Let the set of states be the set of vertices of the graph and each transition (q, I_j, q') be an I_j -labelled edge leaving q and entering q' . If there is a path μ in the graph from q to q' , the number of times an edge e occurs in μ is denoted by $\mu(e)$. If $s \notin \{q, q'\}$ is some node occurring in μ , then the number of edges of μ entering s is equal to the number of edges of μ leaving s . These

conditions can be expressed as integer linear constraints.

$$\begin{aligned}
\sum_{e \text{ leaves } q} \mu(e) - \sum_{e \text{ enters } q} \mu(e) &= 1 \\
\sum_{e \text{ enters } q'} \mu(e) - \sum_{e \text{ leaves } q'} \mu(e) &= 1 \\
s \notin \{q, q'\} : \sum_{e \text{ enters } s} \mu(e) &= \sum_{e \text{ leaves } s} \mu(e)
\end{aligned} \tag{1}$$

Lemma 22 (Theorem 2.1, [20]). *In a directed graph $G = (V, E)$ (possibly with self loops and parallel edges), let $\mu : E \rightarrow \mathbb{N}$ be a function such that the underlying undirected graph induced by edges e such that $\mu(e) > 0$ is connected. Then, there is a path from q to q' with each edge e occurring $\mu(e)$ times iff μ satisfies the constraints (1) above.*

If the beginning and the end of a path are same (i.e., if $q = q'$), small modifications of (1) and Lemma 22 are required. Finally we can prove our desired theorem.

Theorem 23. *Let \mathcal{N} be a 1-safe net with initial marking M_0 and ϕ be a MSO formula. Parameterized by the vertex cover number of $G(\mathcal{N})$ and the size of ϕ , checking whether \mathcal{N} is a model of ϕ is FPT.*

Proof. By Prop. 21, it is enough to check that there is no accepting paths in $\mathcal{A}_{\mathcal{N}} \times \mathcal{A}_{\neg\phi}$ and $\mathcal{A}_{\mathcal{N}} \times \mathcal{B}_{\neg\phi}$. To check the existence of accepting paths in $\mathcal{A}_{\mathcal{N}} \times \mathcal{B}_{\neg\phi}$, we have to check if from some initial state in $\{M_0[\mathcal{S}] \times Q_{0\mathcal{B}}\}$, we can reach some vertex in a maximal strongly connected component induced by \perp -labelled edges, which contains some states from $Q_{\mathcal{N}} \times F_{\mathcal{B}}$. For every such initial state q and a vertex q' in such a strongly connected component, check the feasibility of (1) along with the following constraint for each interface I :

$$\sum_{e \text{ is } I\text{-labelled}} \mu(e) \leq u(I) \tag{2}$$

To check the existence of accepting paths in $\mathcal{A}_{\mathcal{N}} \times \mathcal{A}_{\neg\phi}$, check the feasibility of (1) and (2) for every state q in $\{M_0[\mathcal{S}] \times Q_{0\mathcal{A}}\}$ and every state (P_1, q'') in $F_{\mathcal{N}} \times Q_{\mathcal{A}}$ with some $q_F \in F_{\mathcal{A}}$ such that $(q'', P_1 \cap P_{\phi}, q_F) \in \delta_{\mathcal{A}}$. If some marking M with $M[\mathcal{S} = P_1]$ enables some transition removing a token from some place with interface I , then for each such interface, add the following constraint:

$$\sum_{e \text{ is } I\text{-labelled}} \mu(e) = u(I) \tag{3}$$

The variables in the above ILP instances are $\mu(e)$ for each edge e . The number of variables in each ILP instance is bounded by some function of the parameters. As ILP is FPT when parameterized by the number of variables [13, 14, 11], the result follows. \square

The dependence of the running time of the above algorithm on formula size is non-elementary if the formula is MSO [15]. The dependence reduces to single exponential in case of LTL formulas [22]. The dependence on vertex cover number is dominated by the running time of ILP, which is singly exponential in the number of its variables. The number of variables in turn depends on the number of VC-interfaces (Def. 14). In the worst case, this can be triply exponential but a given 1-safe Petri net need not have all possible VC-interfaces.

5 Conclusion

The main idea behind the FPT upper bound for MSO/LTL model checking is the fact that the problem can be reduced to graph reachability and hence to ILP. It remains to be seen if such techniques or others can be applied for branching time logics such as CTL.

We have some negative results with pathwidth and benefit depth as parameters and a positive result with vertex cover number as parameter. We think it is a challenging problem to identify other parameters associated with 1-safe Petri nets for which standard problems in the concurrency literature are FPT. Another direction for further work, suggested by a referee, is to check if the upper bound can be extended to other classes of Petri nets such as communication-free nets.

The results of Sect. 3 proves hardness for the lowest level of the W-hierarchy. It remains to be seen if the lower bounds could be made tighter. The parameterized classes PARANP and XP include the whole W-hierarchy. Lower bounds or upper bounds corresponding to these classes would be interesting.

References

1. H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Alg.*, 21(2):358–402, 1996.
2. J. R. Büchi. On a decision method in restricted second-order arithmetic. In *Logic, Methodology, Philosophy and Science*, pages 1–11. Stanford Univ Press, 1962.
3. B. Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation*, 85:12–75, 1990.
4. S. Demri, F. Laroussinie, and P. Schnoebelen. A parametric analysis of the state-explosion problem in model checking. *J. Comput. Syst. Sci.*, 72(4):547–575, 2006.
5. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
6. R. G. Downey, M. R. Fellows, and U. Stege. Parameterized complexity: A framework for systematically confronting computational intractability. In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, volume 49 of *DIMACS*, pages 49–100. 1999.
7. D. Drusinsky and D. Harel. On the power of bounded concurrency I: Finite automata. *J. Assoc. Comput. Mach.*, 41(3):517–539, 1994.
8. J. Esparza. *Decidability and complexity of Petri net problems — An introduction*, volume 1491 of *LNCS*, pages 374–428. 1998.
9. M. R. Fellows, D. Lokshantov, N. Misra, F. A. Rosamond, and S. Saurabh. Graph layout problems parameterized by vertex cover. In *ISAAC*, volume 5369 of *LNCS*, pages 294–305, 2008.

10. J. Flum and M. Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.
11. A. Frank and E. Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, January 1987.
12. P. Habermehl. On the complexity of the linear-time μ -calculus for Petri-nets. In *ATPN*, volume 1248 of *LNCS*, pages 102–116, 1997.
13. R. Kannan. Minkowski’s convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.
14. H. W. Lenstra. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8:538–548, 1983.
15. A. R. Meyer. Weak monadic second order theory of successor is not elementary-recursive. In *Proc. Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 132–154. 1975.
16. M. Praveen. Does treewidth help in modal satisfiability? (extended abstract). In *MFCS*, volume 6281 of *LNCS*, pages 580–591, 2010. Full version <http://arxiv.org/abs/1006.2461>.
17. M. Praveen. Small vertex cover makes Petri net coverability and boundedness easier. In *IPEC*, volume 6478 of *LNCS*, pages 216–227, 2010.
18. M. Praveen and K. Lodaya. Modelchecking counting properties of 1-safe nets with buffers in paraspaces. In *FSTTCS*, volume 4 of *LIPICs*, pages 347–358, 2009.
19. C. Rackoff. The covering and boundedness problems for vector addition systems. *Theoret. Comp. Sci.*, 6:223–231, 1978.
20. C. Reutenauer. *The mathematics of Petri nets*. 1990. Translated by I. Craig.
21. M. Samer and S. Szeider. Constraint satisfaction with bounded treewidth revisited. *J. Comput. Syst. Sci.*, 76(2):103–114, 2010.
22. M. Vardi. An automata-theoretic approach to linear temporal logic. In F. Moller and G. Birtwistle, editors, *Logics for Concurrency: Structure versus Automata*, volume 1043 of *LNCS*, pages 238–266. 1996.